TP FIT/IoT-Lab Communication

TP#2 using FIT/IoT-Lab
Lecturer: Keun-Woo Lim
Lecture slides for RIO201
24-10-2018

What to do today

- **Enable communication between devices**
  - HTTP

- **Based on this, do the challenges**

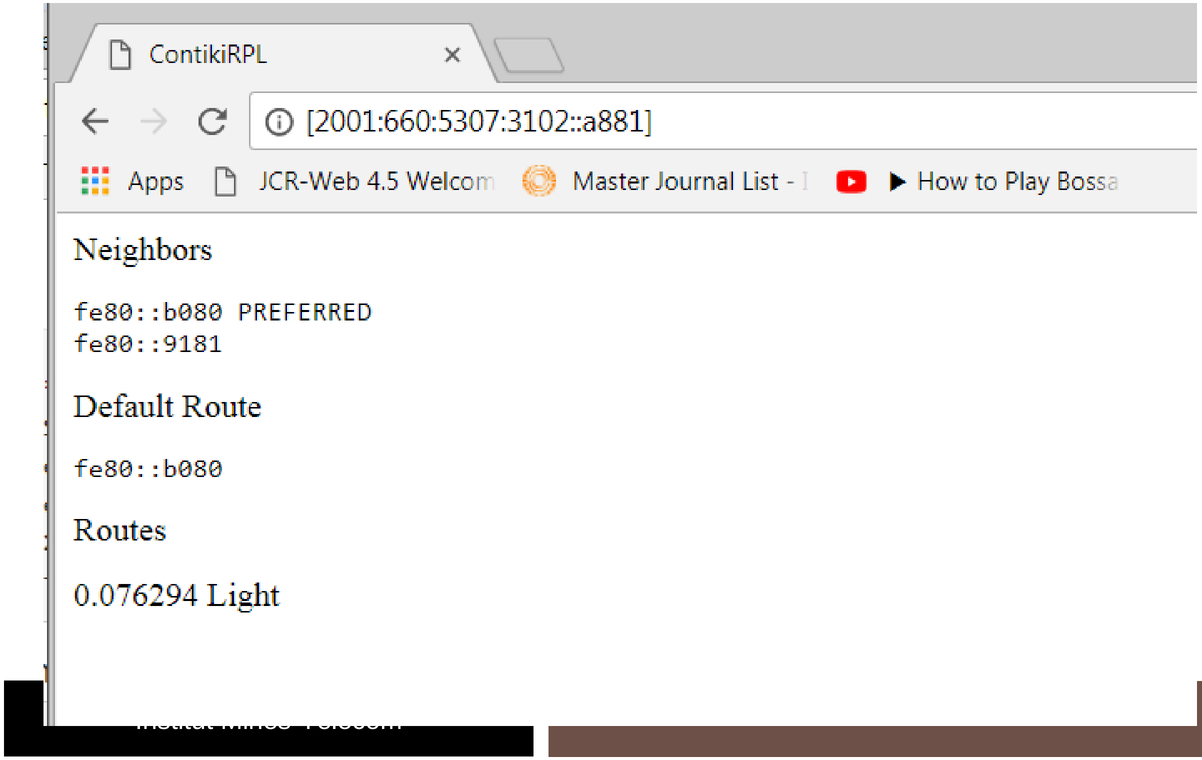
Tutorial – Public IPv6

- **Objective**
  - Create a public HTTP network where you can connect from the Internet
  - Check the function of RPL

- **Let’s try it together!**
  - https://www.iot-lab.info/tutorials/basic-m3-nodes-contiki-uip-stack-with-public-ipv6-on-ssh-front-end/

HTTP tutorial

- **To know you have succeeded,**
  - Open any web browser and put in:
  - http://[2001:660:5307:XXXX::YYYY]
  - XXXX = your subnet, YYYY = one of the HTTP servers



Neighbors

```
fe80::b080 PREFERRED
fe80::9181
```

Default Route

```
fe80::b080
```

Routes

0.076294 Light

Questions here

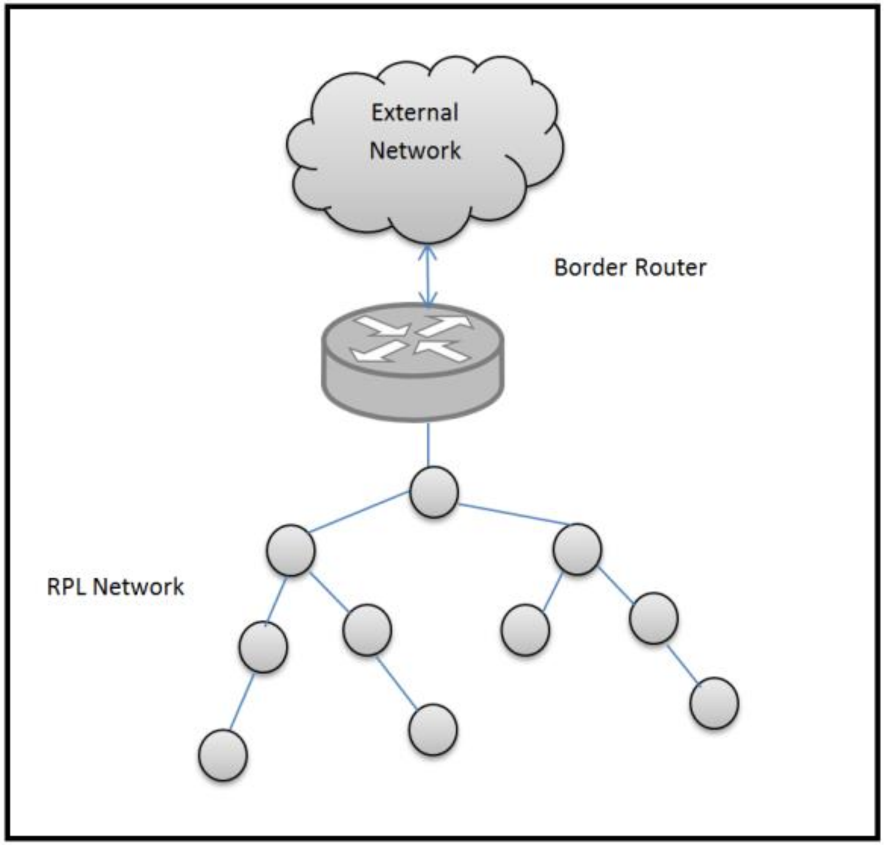
- **What is a Border Router?**
- **What is a HTTP Server?**
- **Why do we need to find an available IPv6 Prefix?**
- **What is a turnslip?**

- **These are all needed for you to connect to the sensor via Internet!!**
  - Makes it seen from outside

- **If there is a HTTP server…**
  - You can see it from a browser!!

Border router

- **What is a Border Router?**
  - Access point to internal and external network



HTTP server

- **An entity that accepts HTTP based requests from the Internet**
  - Based on TCP

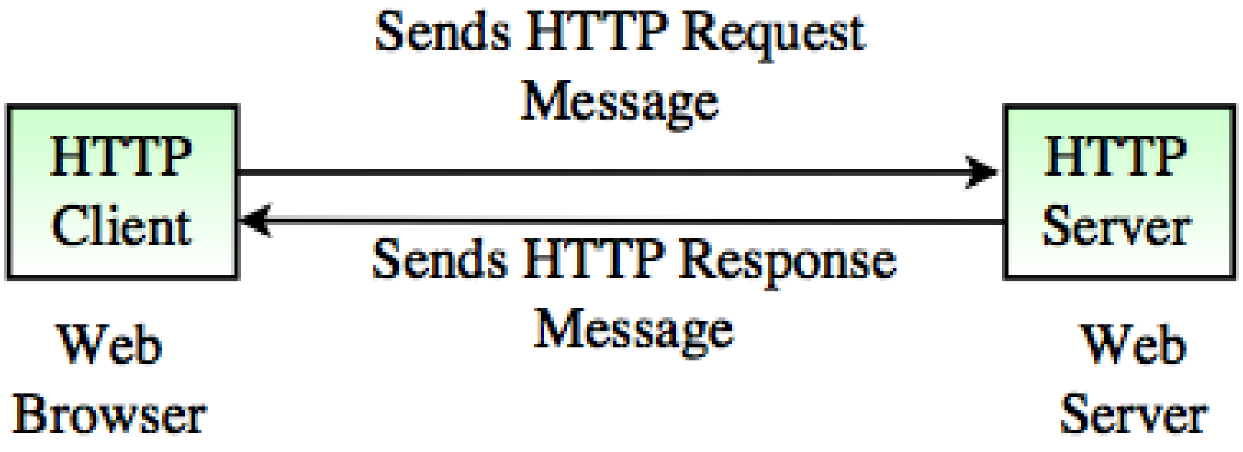
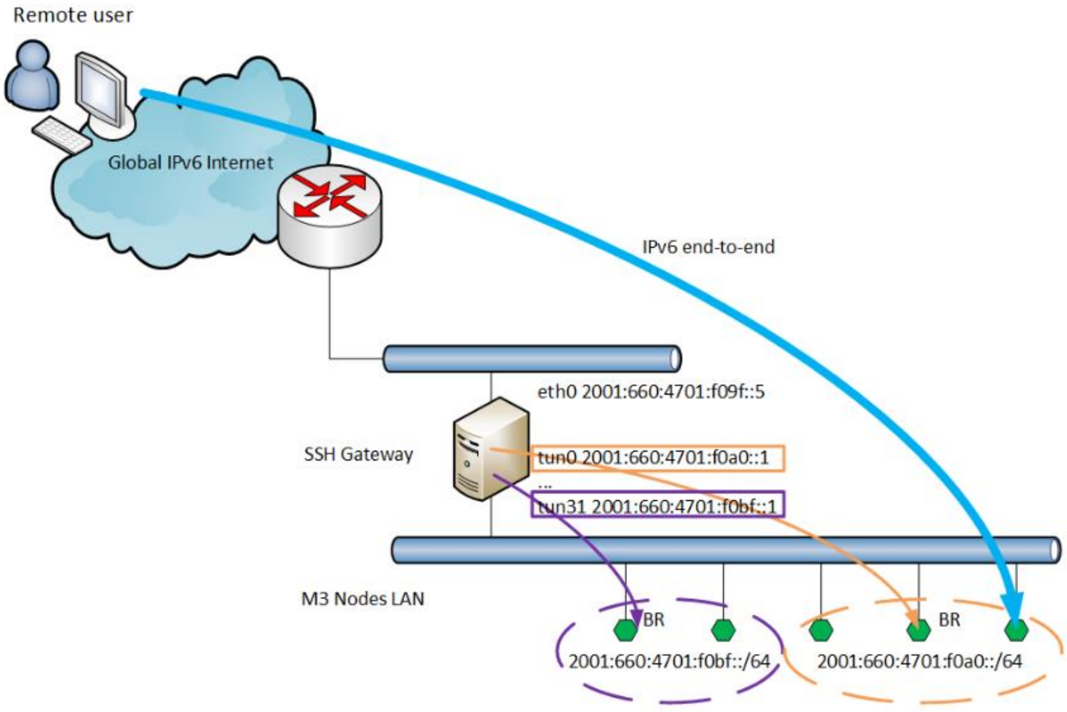


Fig. HTTP Protocol

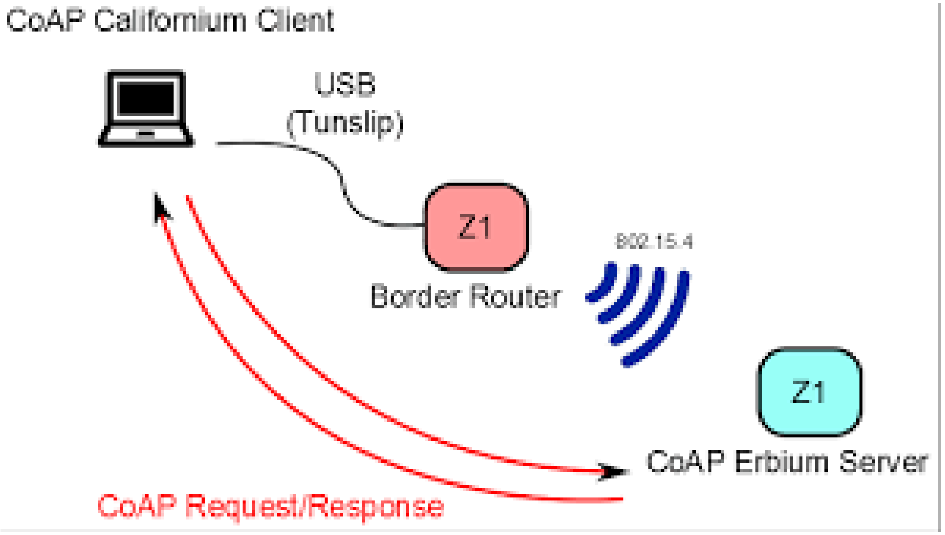
IPv6 Prefixes

- **Needed for groups of entities close to each other, use of prefixes can cluster them and make them easier to find**



Turnslip

- Tool used to bridge IP traffic between a host and another network element, typically a border router, over a serial line.



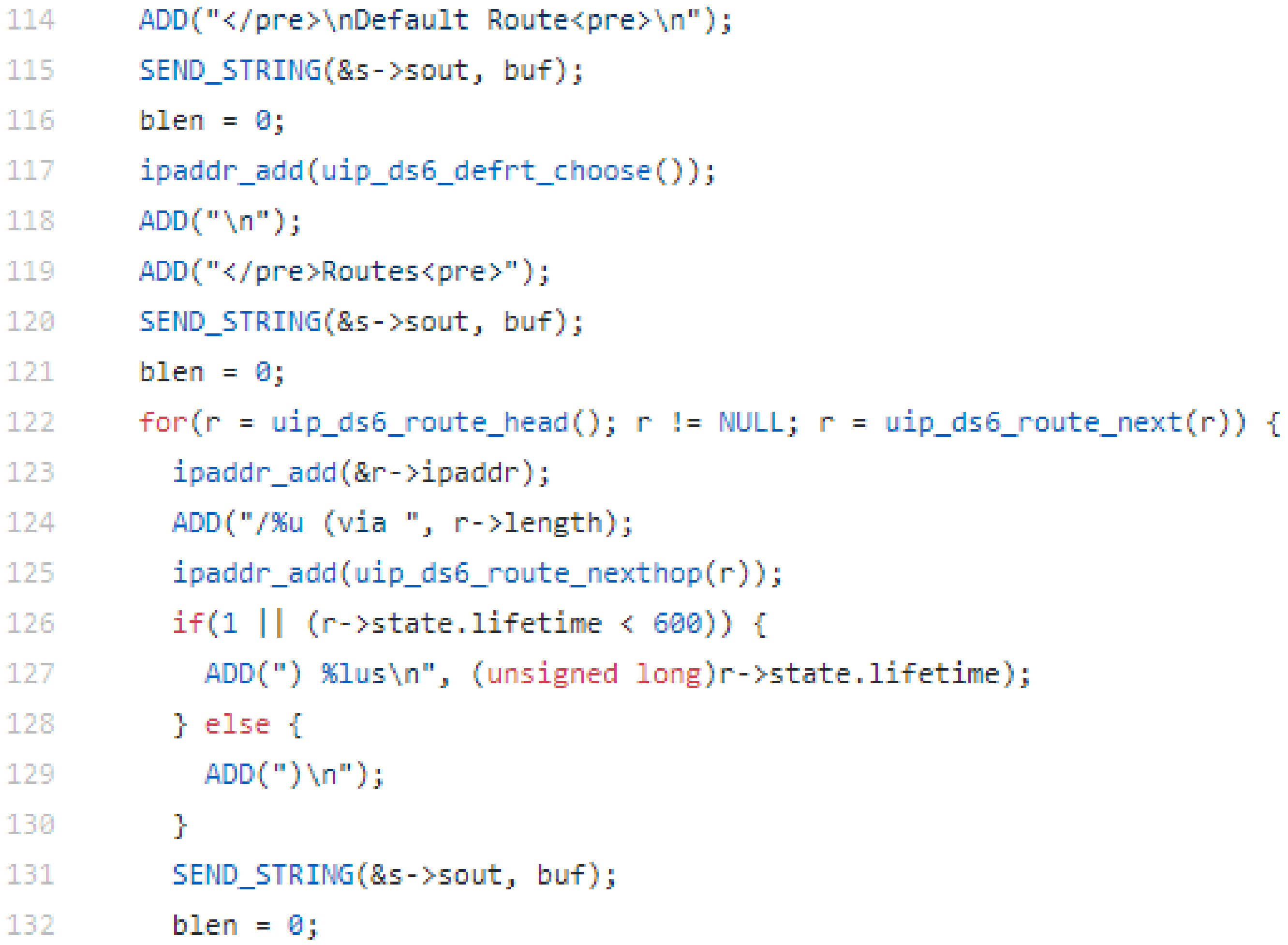
Challenge for today

- **Integrate HTTP and sensor-collecting!**
  - /iot-lab/parts/Contiki/examples/ipv6/http-server
  - /iot-lab/parts/Contiki/examples/iotlab/03-sensors-collecting

- **GOAL**
  - Create sensor readings from the http-server
  - Use your web browser to get sensor readings from the Internet

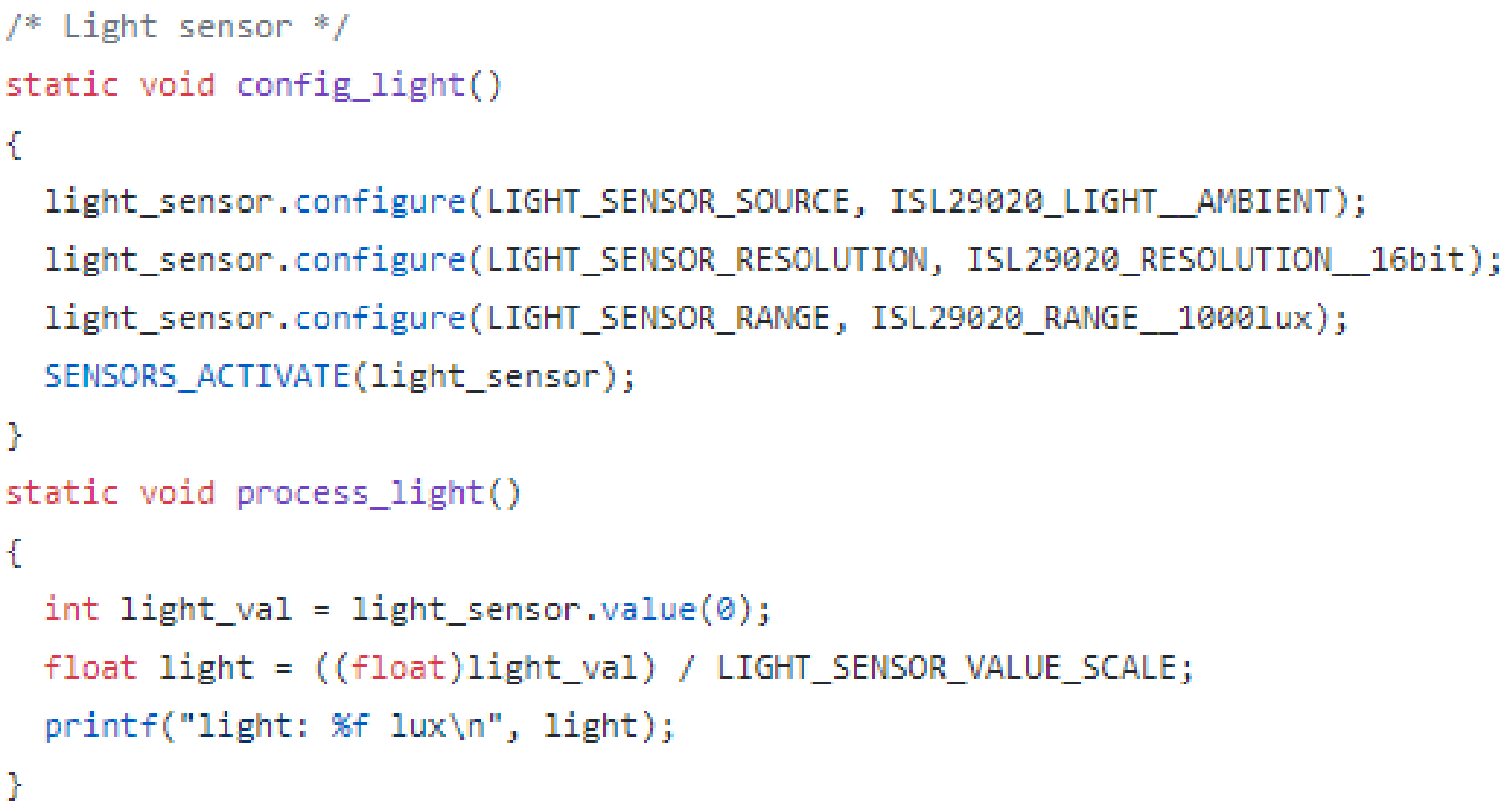
  - For this, let's analyze the http-server code together!

Example of HTTP-server code



```
ADD("</pre>\nDefault Route<pre>\n");
SEND_STRING(&s->sout, buf);
blen = 0;
ipaddr_add(uip_ds6_defrt_choose());
ADD("\n");
ADD("</pre>Routes<pre>");
SEND_STRING(&s->sout, buf);
blen = 0;
for(r = uip_ds6_route_head(); r != NULL; r = uip_ds6_route_next(r)) {
  ipaddr_add(&r->ipaddr);
  ADD("/%u (via ", r->length);
  ipaddr_add(uip_ds6_route_nexthop(r));
  if(1 || (r->state.lifetime < 600)) {
    ADD(") %lus\n", (unsigned long)r->state.lifetime);
  } else {
    ADD(")\n");
  }
  SEND_STRING(&s->sout, buf);
  blen = 0;
```

Example of sensor code



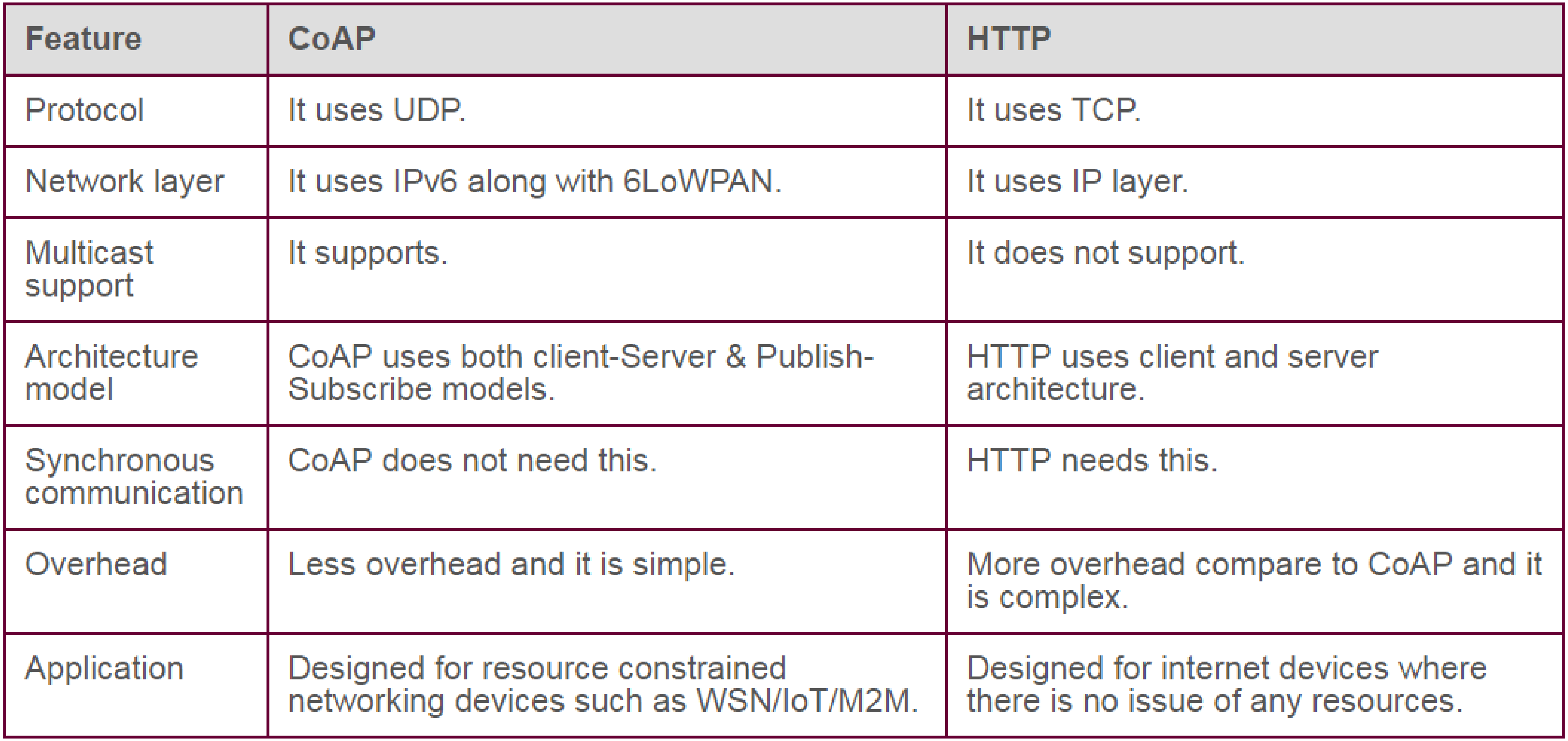
```
/* Light sensor */
static void config_light()
{
  light_sensor.configure(LIGHT_SENSOR_SOURCE, ISL29020_LIGHT__AMBIENT);
  light_sensor.configure(LIGHT_SENSOR_RESOLUTION, ISL29020_RESOLUTION__16bit);
  light_sensor.configure(LIGHT_SENSOR_RANGE, ISL29020_RANGE__1000lux);
  SENSORS_ACTIVATE(light_sensor);
}
static void process_light()
{
  int light_val = light_sensor.value(0);
  float light = ((float)light_val) / LIGHT_SENSOR_VALUE_SCALE;
  printf("light: %f lux\n", light);
}
```

Client / Server connection using CoAP

TP#2 using FIT/IoT-Lab

Why CoAP?

- **CoAP vs HTTP**



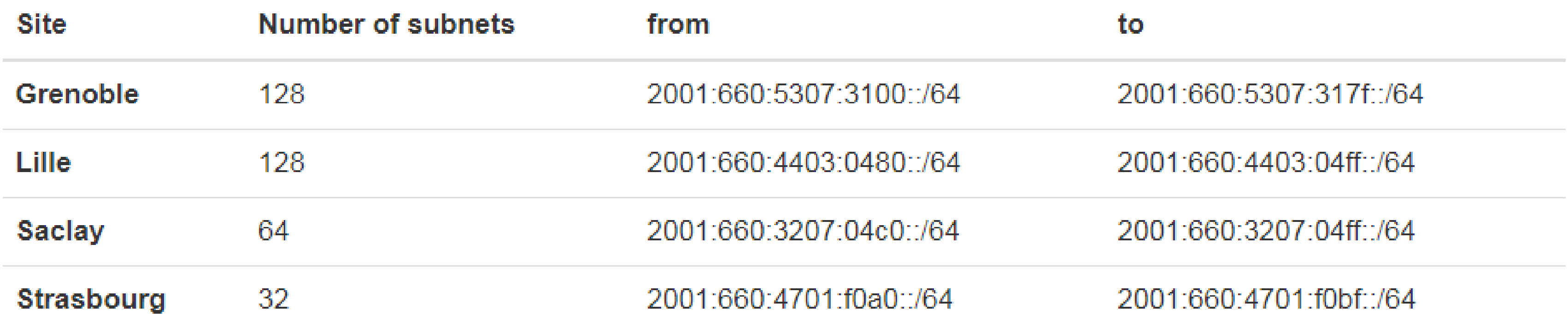
Feature	CoAP	HTTP
Protocol	It uses UDP.	It uses TCP.
Network layer	It uses IPv6 along with 6LoWPAN.	It uses IP layer.
Multicast support	It supports.	It does not support.
Architecture model	CoAP uses both client-Server & Publish-Subscribe models.	HTTP uses client and server architecture.
Synchronous communication	CoAP does not need this.	HTTP needs this.
Overhead	Less overhead and it is simple.	More overhead compare to CoAP and it is complex.
Application	Designed for resource constrained networking devices such as WSN/IoT/M2M.	Designed for internet devices where there is no issue of any resources.

In general,

- CoAP is suited for lightweight IoT devices
- Less overhead, but how can we really see this?

CoAP tutorial

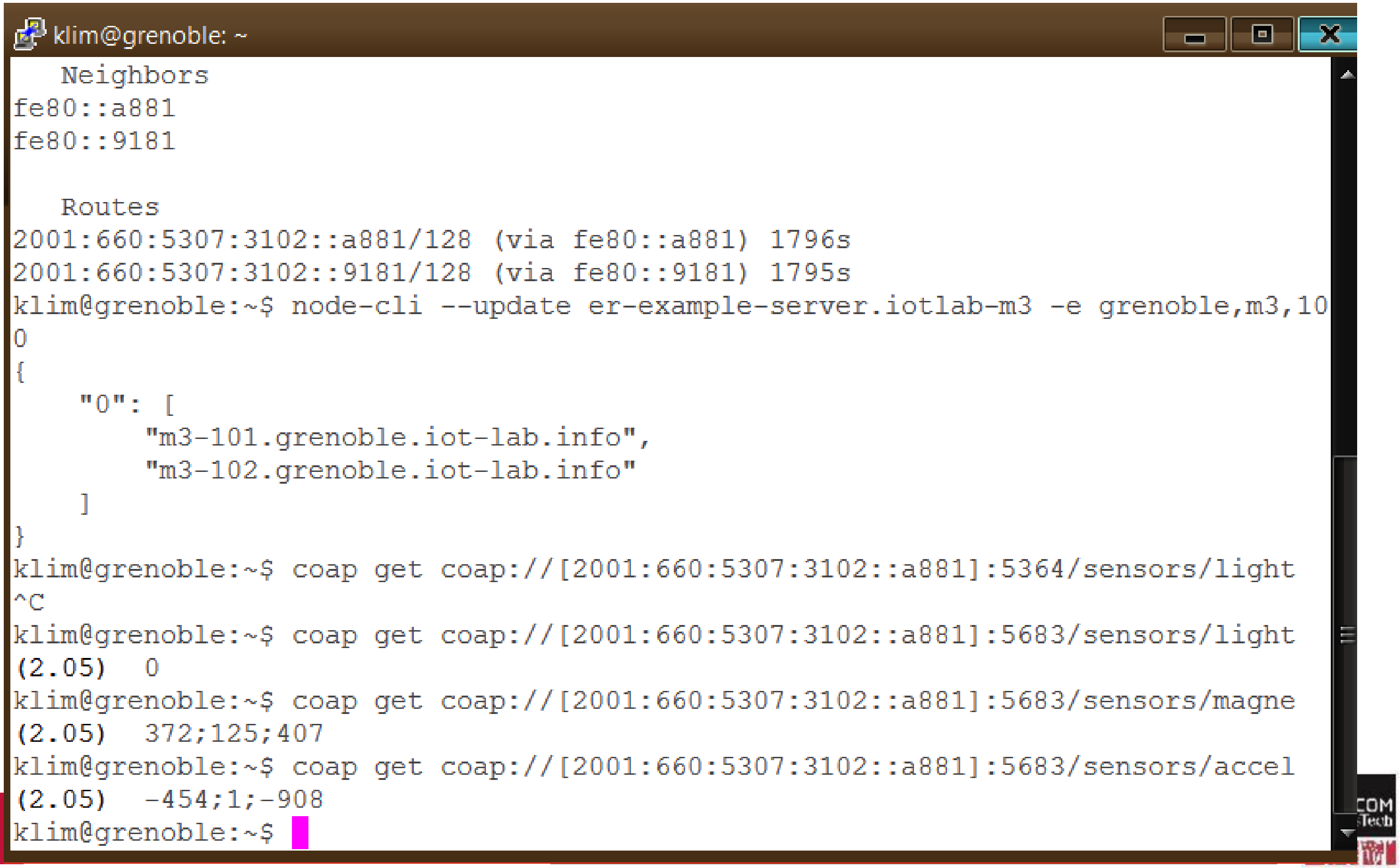
- **We do have the IPv6 subnet for Paris now**
- **For now, only work on 3 sensors**
  - 1 Border router
  - 2 HTTP servers
  - Try to see if you can make a two-hop network



Site	Number of subnets	from	to
Grenoble	128	2001:660:5307:3100::/64	2001:660:5307:317f::/64
Lille	128	2001:660:4403:0480::/64	2001:660:4403:04ff::/64
Saclay	64	2001:660:3207:04c0::/64	2001:660:3207:04ff::/64
Strasbourg	32	2001:660:4701:f0a0::/64	2001:660:4701:f0bf::/64

CoAP tutorial

- **To know you have succeeded,**
  - On the bash command, type

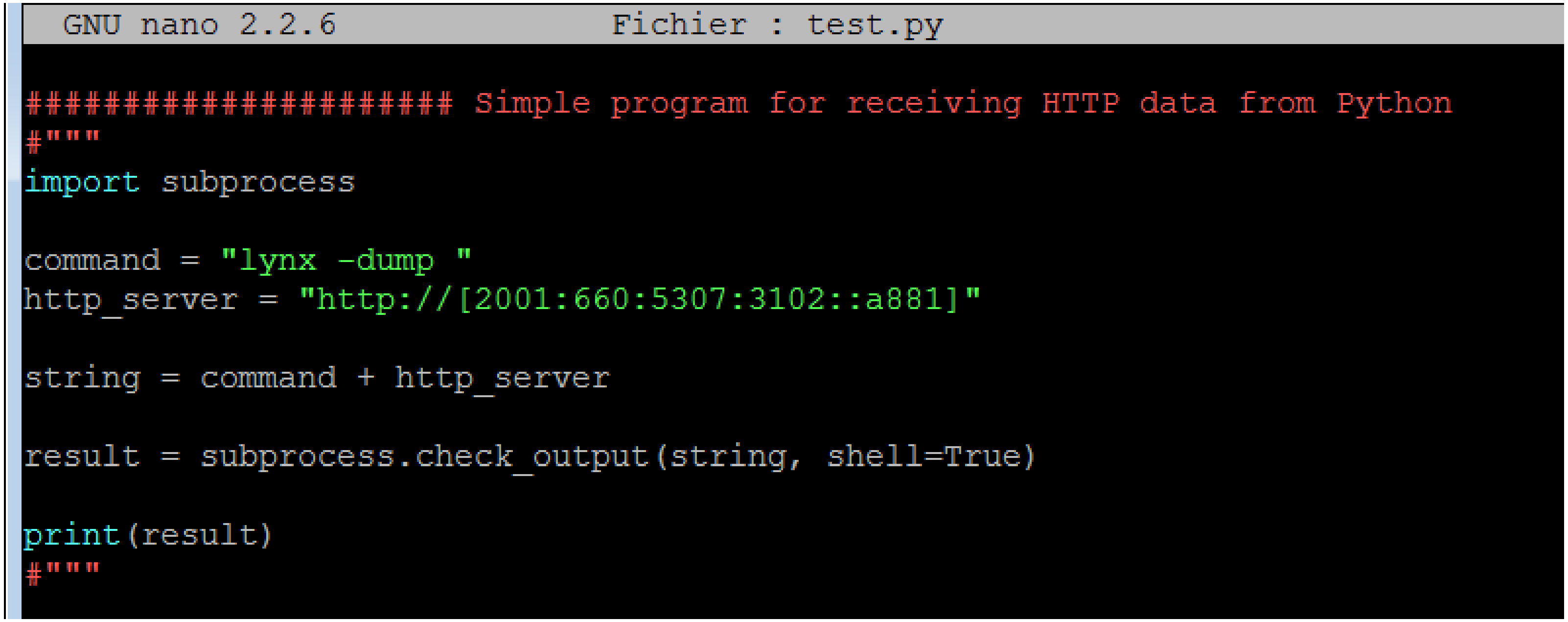


```
   Neighbors
fe80::a881
fe80::9181

   Routes
2001:660:5307:3102::a881/128 (via fe80::a881) 1796s
2001:660:5307:3102::9181/128 (via fe80::9181) 1795s
klim@grenoble:~$ node-cli --update er-example-server.iotlab-m3 -e grenoble,m3,10
0
{
    "0": [
        "m3-101.grenoble.iot-lab.info",
        "m3-102.grenoble.iot-lab.info"
    ]
}
klim@grenoble:~$ coap get coap://[2001:660:5307:3102::a881]:5364/sensors/light
^C
klim@grenoble:~$ coap get coap://[2001:660:5307:3102::a881]:5683/sensors/light
(2.05)   0
klim@grenoble:~$ coap get coap://[2001:660:5307:3102::a881]:5683/sensors/magne
(2.05)   372;125;407
klim@grenoble:~$ coap get coap://[2001:660:5307:3102::a881]:5683/sensors/accel
(2.05)   -454;1;-908
klim@grenoble:~$
```

A program to collect data

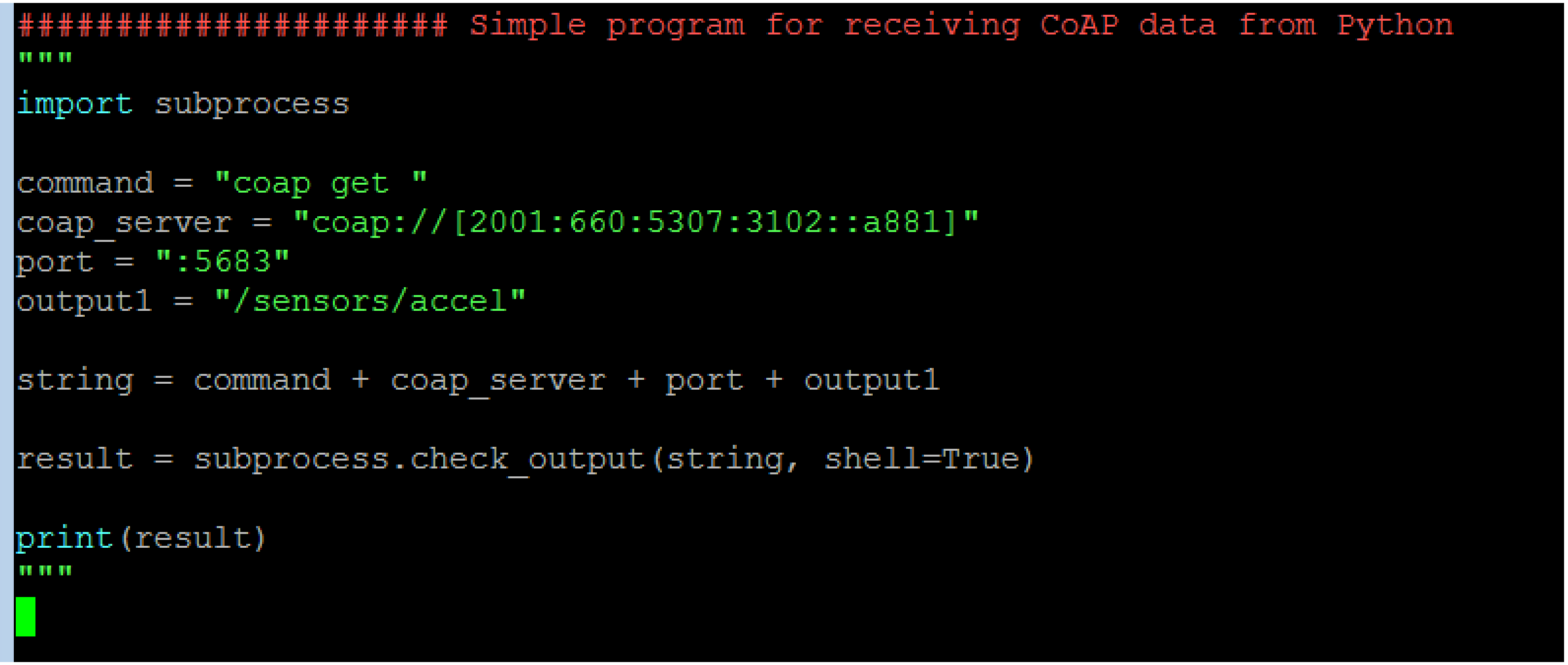
- **Now we know that both HTTP and CoAP servers are public**
  - We can do everything with them
  - Let’s make a program in python to get data from the CoAP server

Example in python for HTTP



```
  GNU nano 2.2.6                Fichier : test.py

###################### Simple program for receiving HTTP data from Python
#"""
import subprocess

command = "lynx -dump "
http_server = "http://[2001:660:5307:3102::a881]"

string = command + http_server

result = subprocess.check_output(string, shell=True)

print(result)
#"""
```

Example in python for CoAP



```
###################### Simple program for receiving CoAP data from Python
"""
import subprocess

command = "coap get "
coap_server = "coap://[2001:660:5307:3102::a881]"
port = ":5683"
output1 = "/sensors/accel"

string = command + coap_server + port + output1

result = subprocess.check_output(string, shell=True)

print(result)
"""
```

Ready for an exercise?

- **Create a python program that collects sensing data from CoAP server every period**
  - Period = one second
  - Data = sensors/gyro, sensors/pressure
  - Node = 1 border router, 2 CoAP servers
  - differentiate information from different nodes